```
@ File:    agarwal2.s
@ Author:  Vasu Agarwal
@ Email: vra0004@uah.edu
@ Class: CS 413-01 Spring 2023
@
@  as -o agarwal2.o agarwal2.s
@  gcc -o agarwal2 agarwal2.o
@ ./agarwal2


.equ READERROR, 0                @Used to check for scanf read error.

.global main
main:

@@@@@@@
prompt:
@@@@@@@
      ldr r0, =strPrompt1
      bl printf

      @scan for user calculation choice
      ldr r0, =charInputPattern @ Setup to read in one number.
      ldr r1, =charInput        @ load r1 with the address of where the input will
be put

      bl  scanf                @ scan the keyboard.

      ldr r1, =charInput        @ Have to reload r1 because it gets wiped out.
      ldr r1, [r1]             @ Read the contents of intInput and store in r1 so
that

@@@@@@@@@@
inputCheck:
@@@@@@@@@@

      @check user input to see what choice was made

      cmp r1, #'1'
      beq promptTriangle

      cmp r1, #'2'
      beq promptRectangle

      cmp r1, #'3'
      beq trapezoidPrompt

      cmp r1, #'4'
      beq squarePrompt

@@@@@@@@
notValid:
@@@@@@@@

      ldr r0, =strInvalid
      bl printf

      b prompt
```

```
@@@@@@@@@@@@@
promptTriangle:
@@@@@@@@@@@@@

base_1:
      ldr r0, =strTriPrompt1  @prompt to enter base
      bl printf

      @scan base
      ldr r0, =numInputPattern @ Setup to read in one number.
      ldr r1, =intInput        @ load r1 with the address of where the
      bl  scanf                @ scan the keyboard.
      cmp r0, #READERROR       @ Check for a read error.
      beq readerror1           @ If there was a read error go handle it.
      ldr r1, =intInput        @ Have to reload r1 because it gets wiped out.
      ldr r7, [r1]

      cmp r7, #0
      ble base_1

height_1:
      ldr r0, =strTriPrompt2  @prompt to enter height
      bl printf

      @scan height
      ldr r0, =numInputPattern @ Setup to read in one number.
      ldr r1, =intInput        @ load r1 with the address of where the
      bl  scanf                @ scan the keyboard.
      cmp r0, #READERROR       @ Check for a read error.
      beq readerror1            @ If there was a read error go handle it.
      ldr r1, =intInput        @ Have to reload r1 because it gets wiped out.
      ldr r8, [r1]

      cmp r8, #0
      ble height_1

      push {r7,r8}
      bl solvTriArea

      pop {r1}

      ldr r0, =strTriResult
      bl printf

      b exitPrompt
@@@@@@@@@@@@@@@
promptRectangle:
@@@@@@@@@@@@@@@

width_1:
      ldr r0, =strRectPrompt1 @prompt to enter width
      bl printf

      @scan width
      ldr r0, =numInputPattern @ Setup to read in one number.
      ldr r1, =intInput        @ load r1 with the address of where the
      bl  scanf                @ scan the keyboard.
      cmp r0, #READERROR       @ Check for a read error.
      beq readerror2            @ If there was a read error go handle it.
```

```
      ldr r1, =intInput        @ Have to reload r1 because it gets wiped out.
      ldr r7, [r1]

      cmp r7, #0
      ble width_1

length_2:
      ldr r0, =strRectPrompt2 @prompt to enter length
      bl printf

      @scan length
      ldr r0, =numInputPattern @ Setup to read in one number.
      ldr r1, =intInput        @ load r1 with the address of where the
      bl  scanf                @ scan the keyboard.
      cmp r0, #READERROR       @ Check for a read error.
      beq readerror2            @ If there was a read error go handle it.
      ldr r1, =intInput        @ Have to reload r1 because it gets wiped out.
      ldr r8, [r1]

      cmp r8, #0
      ble length_2

      push {r7,r8}
      bl solvRectArea

      pop {r1}

      ldr r0, =strRectResult
      bl printf

      b exitPrompt
@@@@@@@@@@@@@@@@
trapezoidPrompt:
@@@@@@@@@@@@@@@@

baseA:
      ldr r0, =strTrapPrompt1 @prompt to enter baseA
      bl printf

      @scan baseA
      ldr r0, =numInputPattern @ Setup to read in one number.
      ldr r1, =intInput        @ load r1 with the address of where the
      bl  scanf                @ scan the keyboard.
      cmp r0, #READERROR       @ Check for a read error.
      beq readerror3            @ If there was a read error go handle it.
      ldr r1, =intInput        @ Have to reload r1 because it gets wiped out.
      ldr r7, [r1]

      cmp r7, #0
      ble baseA

baseB:
      ldr r0, =strTrapPrompt2 @prompt to enter baseB
      bl printf

      @scan baseA
      ldr r0, =numInputPattern @ Setup to read in one number.
      ldr r1, =intInput        @ load r1 with the address of where the
      bl  scanf                @ scan the keyboard.
```

```
      cmp r0, #READERROR       @ Check for a read error.
      beq readerror3            @ If there was a read error go handle it.
      ldr r1, =intInput        @ Have to reload r1 because it gets wiped out.
      ldr r8, [r1]

      cmp r8, #0
      ble baseB


height_3:
      ldr r0, =strTrapPrompt3 @prompt to enter height
      bl printf

      @scan height
      ldr r0, =numInputPattern @ Setup to read in one number.
      ldr r1, =intInput        @ load r1 with the address of where the
      bl  scanf                @ scan the keyboard.
      cmp r0, #READERROR       @ Check for a read error.
      beq readerror3            @ If there was a read error go handle it.
      ldr r1, =intInput        @ Have to reload r1 because it gets wiped out.
      ldr r9, [r1]

      cmp r9, #0
      ble height_3

      push {r7,r8,r9}
      bl solvTrapArea

      pop {r1}

      ldr r0, =strTrapResult
      bl printf

      b exitPrompt
@@@@@@@@@@@@@
squarePrompt:
@@@@@@@@@@@@@

      ldr r0, =strSquarePrompt1    @prompt to enter side length
      bl printf

      @scan side length
      ldr r0, =numInputPattern @ Setup to read in one number.
      ldr r1, =intInput        @ load r1 with the address of where the
      bl  scanf                @ scan the keyboard.
      cmp r0, #READERROR       @ Check for a read error.
      beq readerror4            @ If there was a read error go handle it.
      ldr r1, =intInput        @ Have to reload r1 because it gets wiped out.
      ldr r7, [r1]

      cmp r7, #0
      ble squarePrompt

      push {r7}
      bl solvSquareArea

      pop {r1}

      ldr r0, =strSquareResult
```

```
      bl printf

      b exitPrompt
@@@@@@@@@@@@
solvTriArea:
@@@@@@@@@@@@

      @formula to use, A = (1/2) * height * base
      pop {r7, r8}      @r7 is height, r8 is base

      umull r6, r3, r7, r8
      cmp r3, #0
      bne overflowPrompt

      mov r5, r6, asr #1

      push {r5}

      mov pc, lr

@@@@@@@@@@@@
solvRectArea:
@@@@@@@@@@@@

      @formula to use, A = width * length
      pop {r7, r8}      @r7 is width, r8 is length

      umull r6, r3, r7, r8
      cmp r3, #0
      bne overflowPrompt

      push {r6}

      mov pc, lr

@@@@@@@@@@@@
solvTrapArea:
@@@@@@@@@@@@

      @formula to use, A = (1/2) * (baseA + baseB) * height
      pop {r7, r8, r9}  @r7 is baseA, r8 is baseB, r9 is height



      add r7, r7, r8

      umull r6, r3, r7, r9
      cmp r3, #0
      bne overflowPrompt

      mov r5, r6, asr #1

      push {r5}

      mov pc, lr

@@@@@@@@@@@@@@@
solvSquareArea:
@@@@@@@@@@@@@@@
```

```
      @formula to use, A = side * side
      pop {r7}    @r7 is side

      umull r6, r3, r7, r7
      cmp r3, #0
      bne overflowPrompt

      push {r6}

      mov pc, lr

@@@@@@@@@@@@@@
overflowPrompt:
@@@@@@@@@@@@@@

      ldr r0, =strOverflowResult    @print out overflow prompt
      bl printf

      b prompt                @go back to prompt

@@@@@@@@@@
exitPrompt:
@@@@@@@@@@

      ldr r0, =strContPrompt
      bl printf

      ldr r0, =charInputPattern @ Setup to read in one number.
      ldr r1, =charInput        @ load r1 with the address of where the input will
be put
      bl  scanf                @ scan the keyboard.
      ldr r1, =charInput        @ Have to reload r1 because it gets wiped out.
      ldr r1, [r1]             @ Read the contents of intInput and store in r1 so
that

      cmp r1, #'Y'
      beq prompt

      cmp r1, #'N'
      beq exit

      b exitPrompt



@@@@@@@@@@
readerror1:
@@@@@@@@@@

      @ Got a read error from the scanf routine. Clear out the input buffer then
      @ branch back for the user to enter a value.
      @ Since an notValid entry was made we now have to clear out the input buffer
by
      @ reading with this format %[^\n] which will read the buffer until the user
      @ presses the CR.

      ldr r0, =strInputPattern
      ldr r1, =strInputError   @ Put address into r1 for read.
```

```
      bl scanf                @ scan the keyboard.

@  Not going to do anything with the input. This just cleans up the input buffer.
@  The input buffer should now be clear so get another input.

      b promptTriangle

@@@@@@@@@@
readerror2:
@@@@@@@@@@

      @ Got a read error from the scanf routine. Clear out the input buffer then
      @ branch back for the user to enter a value.
      @ Since an notValid entry was made we now have to clear out the input buffer
by
      @ reading with this format %[^\n] which will read the buffer until the user
      @ presses the CR.

      ldr r0, =strInputPattern
      ldr r1, =strInputError   @ Put address into r1 for read.
      bl scanf                 @ scan the keyboard.
@  Not going to do anything with the input. This just cleans up the input buffer.
@  The input buffer should now be clear so get another input.

      b promptRectangle

@@@@@@@@@@
readerror3:
@@@@@@@@@@

      @ Got a read error from the scanf routine. Clear out the input buffer then
      @ branch back for the user to enter a value.
      @ Since an notValid entry was made we now have to clear out the input buffer
by
      @ reading with this format %[^\n] which will read the buffer until the user
      @ presses the CR.

      ldr r0, =strInputPattern
      ldr r1, =strInputError   @ Put address into r1 for read.
      bl scanf                 @ scan the keyboard.
@  Not going to do anything with the input. This just cleans up the input buffer.
@  The input buffer should now be clear so get another input.

      b trapezoidPrompt

@@@@@@@@@@
readerror4:
@@@@@@@@@@

      @ Got a read error from the scanf routine. Clear out the input buffer then
      @ branch back for the user to enter a value.
      @ Since an notValid entry was made we now have to clear out the input buffer
by
      @ reading with this format %[^\n] which will read the buffer until the user
      @ presses the CR.

      ldr r0, =strInputPattern
      ldr r1, =strInputError   @ Put address into r1 for read.
      bl scanf                 @ scan the keyboard.
```

```
@  Not going to do anything with the input. This just cleans up the input buffer.
@  The input buffer should now be clear so get another input.

      b squarePrompt

@@@@@
exit:
@@@@@

@ Force the exit of this program and return command to OS.

      mov  r7, #0X01
      svc  0



.data

@prompts
.balign 4
strPrompt1: .asciz "Welcome to Lab2! Please choose what you'd like to calculate the
area for.\n 1 - Triangle \n 2 - Rectangle \n 3 - Trapezoid \n 4 - Square \n"

.balign 4
strInvalid: .asciz "Invalid Input, try again\n"

.balign 4
strTriPrompt1: .asciz "Triangle requires base and height. \nEnter the base: \n"

.balign 4
strTriPrompt2: .asciz "Enter height: \n"

.balign 4
strRectPrompt1: .asciz "Rectangle requires width and length. \nEnter the width: \n"

.balign 4
strRectPrompt2: .asciz "Enter length: \n"

.balign 4
strTrapPrompt1: .asciz "Trapezoid requires baseA, baseB, and height. \nEnter the
baseA: \n"

.balign 4
strTrapPrompt2: .asciz "Enter baseB: \n"

.balign 4
strTrapPrompt3: .asciz "Enter height: \n"

.balign 4
strSquarePrompt1: .asciz "Square requires side length. \n Enter the side length: \
n"

.balign 4
strContPrompt: .asciz "Would you like to continue? Y/N\n"

.balign 4
onehalf: .word 1/2

@calc result patterns
```

```
.balign 4
strTriResult: .asciz "The triangle's area is %d.\n"

.balign 4
strRectResult: .asciz "The rectangle's area is %d.\n"

.balign 4
strTrapResult: .asciz "The trapazoid's area is %d.\n"

.balign 4
strSquareResult: .asciz "The square's area is %d.\n"

.balign 4
strOverflowResult: .asciz "There was an overflow.\n"


@scanf Patterns

.balign 4
charInputPattern: .asciz "%s"  @ integer format for read.

.balign 4
charInput: .word 0   @ Location used to store the user input.

.balign 4
numInputPattern: .asciz "%d"  @ integer format for read.

.balign 4
intInput: .word 0   @ Location used to store the user input.

.balign 4
strInputPattern: .asciz "%[^\n]" @ Used to clear the input buffer for notValid
input.

.balign 4
strInputError: .skip 100*4  @ User to clear the input buffer for notValid input.

.global printf

.global scanf


@end of code and end of file. Leave a blank line after this.
```